Firm Docket No. 1[redacted].005
WindRiver Ref. No. 2000.021

Title of Application: Unified Exception Handling For Hierarchical Multi-Interrupt Architectures

Inventor: Kenneth J. Brenner, Jr.
Address: 266 North River Road,
Milford, NH 03055 (U.S. citizen)

Inventor: Richard E. Carter, Jr.
Address: 74 Talent Road
Litchfield, NH 03052 (U.S. citizen)

Sampson & Associates, P.C.
50 Congress Street
Boston, Massachusetts 02109

Unified Exception Handling For Hierarchical Multi-Interrupt Architectures

BACKGROUND INFORMATION

The skilled artisan will recognize that in microprocessor operation, exceptions arise when there is a need for the normal flow of program execution to be broken, for example, so that the processor can be diverted to handle an interrupt from a peripheral. The processor state just prior to handling the exception must be preserved so that the original program can be resumed when the exception routine has completed. Many exceptions may arise at the same time.

One common type of microprocessor is sold under the trademark ARM®, owned by Advanced RISC Machines, Ltd. (Cambridge, UK) based on an architecture set forth in the *ARM Architecture Reference Manual*, Version B (*et. seq.*), dated 26 July 1996, pages 2-1 to 2-10 of which are incorporated by reference herein. (A product line of such ARM® type processors, including the SA-110™ microprocessor, is available from INTEL® Corporation (San Jose, CA). Such ARM® processors handle exceptions by making use of banked registers to save the processor

state. The old program counter (PC) and Current Processor Status Register (CPSR) contents are copied into appropriate registers (e.g., R14 and SPSR), and the PC and mode bits in the CPSR bits are forced to a value that depends on the exception. Interrupt disable flags may be set to try and prevent otherwise unmanageable nestings of exceptions. For example, upon receiving an interrupt, the interrupt disable flags may be set to prevent new interrupts from being accepted until the current interrupt has been processed.

These ARM® processors separate interrupt handling into discrete instruction streams, i.e., one for IRQs (Interrupt Requests) and another for FIQs (Fast Interrupt Requests). The ARM® architecture generally contemplates that FIQs are to be used for relatively high priority interrupts, such as memory refresh, that are intended to run beneath the Operating System (OS). As such, FIQs are provided with a higher priority than IRQs, and may interrupt the processing of a pending IRQ exception call. In practice, however, it has been found by the inventors of the present invention that users often use FIQs as a part of a multi-tiered interrupt system usable by the OS. Disadvantageously, such use often results in complex nested interrupt stacks which the OS has difficulty, or is unable, to 'unwind' and properly process these interrupts.

As a result, the OS is often unable to support calls from both IRQ and FIQ devices.

Thus, a need exists for an improved method and system for providing a single interrupt instruction stream for both multiple (e.g., IRQ and FIQ) exceptions.

SUMMARY

An embodiment of this invention includes a method of configuring an operating system for an embeddable processor having multiple interrupt types. The method includes the steps of providing an interrupt vector table, which has a first vector address executable upon receipt of an interrupt request of a first type, and a second vector address executable upon receipt of an interrupt request of a second type. The first vector address precedes the second vector address in the vector table. A common interrupt dispatcher is provided. An instruction is inserted into the first vector address that disables the second interrupt mode. At the second vector address, an other instruction is inserted that branches to the common interrupt dispatcher. The common interrupt dispatcher is provided with an interrupt routine that processes the interrupt, and then re-enables the second interrupt type. Interrupt requests are then processed by the common interrupt dispatcher without interruption.

Another aspect of the invention includes a method for configuring an operating system for ARM® processors. The method includes the steps of providing a common interrupt dispatcher, and inserting an instruction into the IRQ vector address that disables the FIQ interrupt mode. An other instruction that branches to the common interrupt dispatcher is inserted at the FIQ vector address. The common interrupt dispatcher is provided with an interrupt routine that processes the interrupt, and then re-enables the FIQ interrupt mode.

A further aspect of the invention includes method of operating a processor. This method includes the steps of providing a common interrupt dispatcher having an interrupt routine that checks a mode identifier to determine whether a received interrupt was of a first or second type, and processes the interrupt. Additional steps include inserting into a first vector address of an interrupt vector table, an instruction that disables subsequent interrupts. An other instruction is inserted at a second vector address of the exception vector table, that branches to the common interrupt dispatcher. An interrupt of the first type is received, followed by branching to the first vector address, and setting the mode identifier to indicate an interrupt of the first type was received. The instruction to disable the first and

second interrupts is executed, and the other instruction is executed to branch to the common interrupt dispatcher. The interrupt is processed by the common interrupt dispatcher without interruption, and the first and second interrupts are re-enabled.

A still further aspect of the invention includes a method of operating a processor. This method includes the steps of providing a common interrupt dispatcher having an interrupt routine that checks a mode identifier to determine whether a received interrupt call was of a first or second interrupt type, and processes both interrupt types. Additional steps include inserting at a second interrupt address in an interrupt vector table, an instruction that branches to the common interrupt dispatcher. Further steps include receiving an interrupt of the second interrupt type, branching to the second vector address, and setting the mode identifier to indicate an interrupt of the second type was received. The instruction to branch to the common interrupt dispatcher is executed, and the interrupt is processed by the common interrupt dispatcher without interruption. to here

Another aspect of the present invention includes an operating system for a processor. The operating system includes. The system further includes a common interrupt

dispatcher, and an instruction in a first vector address executable upon receipt of an IRQ interrupt request, that disables the IRQ and FIQ interrupt modes. An other instruction is located at a second vector address executable upon receipt of a second interrupt request, that branches to the common interrupt dispatcher. The common interrupt dispatcher has an interrupt routine that checks the mode identifier to determine whether a received interrupt call was of the first or second interrupt types, processes the interrupt, and then re-enables the first and second interrupt types. The IRQ and FIQ interrupts are processed by the common interrupt dispatcher without interruption.

Another aspect of the invention includes an article of manufacture. The article of manufacture includes a computer usable medium having a computer readable program code embodied therein. The computer usable medium includes computer readable program code for providing a common interrupt dispatcher for a processor having a first interrupt mode and a second interrupt mode to respectively accept interrupt requests of first and second types, the second interrupt mode having a higher priority than the first interrupt mode, both first and second interrupt modes being disableable to selectively reject interrupt requests of the first and second interrupt types. The

processor also includes a mode status indicator to indicate the current mode of the processor, and an interrupt vector table having a first vector address executable upon receipt of an interrupt request of the first type, and a second vector address executable upon receipt of an interrupt request of the second type, the first vector address preceding the second vector address in the vector table. The processor is adapted to execute at least one instruction in the interrupt vector table without interruption, upon receipt of an interrupt request. Computer readable program code is also provided for inserting an instruction into the first vector address that disables the first and second interrupt modes. Computer readable program code is also included for inserting an other instruction at the second vector address, that branches to the common interrupt dispatcher. This embodiment further includes computer readable program code for providing the common interrupt dispatcher with an interrupt routine that checks the mode identifier to determine whether a received interrupt call was a first interrupt type or a second interrupt type, processes the interrupt, and then re-enables the first and second interrupt modes. Interrupt requests are processed by the common interrupt dispatcher without interruption.

A yet further aspect of the invention includes

computer readable program code for merging tiered interrupts into a unified interrupt handling system for an operating system of an embeddable processor. The computer readable program code includes computer readable program code for providing a common interrupt dispatcher for a processor having a first interrupt mode and a second interrupt mode to respectively accept interrupt requests of first and second types, the second interrupt mode having a higher priority than the first interrupt mode, both first and second interrupt modes being disableable to selectively reject interrupt requests of the first and second interrupt types. The processor also includes a mode status indicator to indicate the current mode of the processor, and an interrupt vector table having a first vector address executable upon receipt of an interrupt request of the first type, and a second vector address executable upon receipt of an interrupt request of the second type, the first vector address preceding the second vector address in the vector table. The processor is adapted to execute at least one instruction in the interrupt vector table without interruption, upon receipt of an interrupt request. This aspect also includes computer readable program code for inserting an instruction into the first vector address that disables the first and second interrupt modes, and computer

readable program code for inserting an other instruction at the second vector address, that branches to the common interrupt dispatcher. Computer readable program code is also included for providing the common interrupt dispatcher with an interrupt routine that checks the mode identifier to determine whether a received interrupt call was a first interrupt type or a second interrupt type, processes the interrupt, and then re-enables the first and second interrupt modes. Interrupt requests are processed by the common interrupt dispatcher without interruption.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other features and advantages of this invention will be more readily apparent from a reading of the following detailed description of various aspects of the invention taken in conjunction with the accompanying drawings, in which:

Fig. 1 is a functional block diagram of prior art exception handling in an ARM® processor; and

Fig. 2 is a functional block diagram of exception handling in an ARM® processor according to the present invention.

DETAILED DESCRIPTION

Referring to the figures set forth in the accompanying Drawings, the illustrative embodiments of the present invention will be described in detail hereinbelow. For clarity of exposition, like features shown in the accompanying Drawings shall be indicated with like reference numerals and similar features as shown in alternate embodiments in the Drawings shall be indicated with similar reference numerals.

An embodiment of the invention includes an instruction sequence and method for processors that implement ARM® architecture (and similar multiple interrupt mode architectures) as described herein. This embodiment of the present invention allows users of such processor architectures to merge multiple interrupt types, such as Fast Interrupt (FIQ) exceptions and normal Interrupt (IRQ) exceptions, into a single exception handling instruction stream, while still allowing the processor to discriminate between the two interrupt sources. The embodiment(s) of the present invention may be implemented as part of a Board Support Package (BSP) for such processors. The embodiment(s) may also be implemented as part of the kernel of an operating system for such processors.

Turning to Fig. 1, previous approaches separate

interrupt handling by use of discrete addresses within exception vector table 10. Address 12 is used for IRQ exceptions, while address 14 is used for FIQ exceptions. As shown, IRQ and FIQ exceptions 11 and 13, generated either internally (i.e., by the processor) or externally (i.e., by hardware peripherals), branch program execution to addresses 12 and 14, respectively. Vector addresses 12 and 14 typically include branch instructions that put the address of discrete IRQ and FIQ Handlers (16 and 18, respectively) in the program counter, to branch program execution thereto. As mentioned hereinabove, in the ARM® architecture, FIQs are given a higher priority than IRQs. As such, the IRQ instruction stream 17 (i.e., between IRQ vector 12 and Interrupt Handler 16) is interruptible by subsequent FIQ exceptions. Disadvantageously, this arrangement may result in the inability of operating systems to support calls from both IRQ and FIQ devices. For example, this aspect may generate complex nested exception routines, which tend to be problematic in many applications, such as those in which FIQs are used by the OS to perform routine bookkeeping functions. Attempts to merge these two streams may lead to an internal race condition within the processor, making it difficult or impossible for the operating system to properly process.

As mentioned hereinabove, many commonly available

ARM® processors handle exceptions by making use of banked registers to save state. Interrupt disable flags may be set to try and prevent otherwise unmanageable nestings of exceptions. When multiple exceptions arise simultaneously, a fixed priority determines the order in which they are handled. For example, in the SA-110™ processor, the priority is as follows: 1) Reset (highest priority); 2) Data abort; 3) FIQ; 4) IRQ; 5) Prefetch abort; and 6) Undefined instruction, software interrupt (lowest priority).

This embodiment of the present invention makes use of specific architecture features to merge the two interrupt streams into a single handler, while unambiguously identifying the source of each interrupt. Turning now to Fig. 2, this embodiment of the present invention includes a single instruction referred to as a Move Status Register (MSR) instruction 20. This instruction 20 serves to simultaneously disable both FIQs and IRQs upon receipt of an IRQ exception. This instruction 20 is inserted at IRQ vector address 12. Thus, upon receipt of an IRQ, i.e., when the PC branches to IRQ vector 12, the MRS instruction 20 disables any subsequent interrupts (both IRQ and FIQ). Instruction 20 accomplishes this by inserting an F bit in the Current Processor Status Register (CPSR) to disable FIQs, and by leaving the I bit set (the I bit is

automatically set upon a branch to the IRQ vector 12) or re-setting the I bit.

The skilled artisan will recognize that in conventional ARM® processors, the "I" bit is set in the CPSR when an IRQ occurs. This blocks any further IRQs until the current IRQ has been processed. The "F" bit is typically set when an FIQ occurs. This blocks further FIQs until the current FIQ has been processed. Since FIQs take priority over IRQs, both IRQs and FIQs are effectively blocked during processing of an FIQ.

This embodiment of the present invention also includes a branch instruction 22, which instructs the processor to branch to an address of a common dispatcher 24 (discussed hereinbelow). Instruction 22 is placed at FIQ vector address 14 of the exception table 10. As shown, the IRQ vector address 12 precedes the FIQ vector address 14 in conventional ARM® exception vector tables. As such, absent a branch instruction or other exception, such as a reset, instruction execution will generally advance from address 12 to address 14.

As mentioned hereinabove, the ARM® architecture sets mode bits within the CPSR that correspond to the type of exception, i.e., either IRQ or FIQ. The architecture also ensures that the first instruction at the vector table is executed without interruption. This embodiment utilizes

this latter feature to effectively ensure that the inserted MRS instruction 20 is properly executed.

After executing the first inserted instruction 20 located at vector address 12, execution falls through to the next instruction address, which in the embodiment shown, is at FIQ vector address 14. Branch instruction 22 previously inserted at this address 14 then instructs the processor to branch to common dispatcher 24.

Alternatively, in the event an FIQ exception 13 is received, the PC will branch directly to FIQ address 14, which in turn, will branch to the common dispatcher 24.

This embodiment of the present invention thus effectively provides a common, or merged, instruction stream at dispatcher 24 for both IRQ and FIQ interrupts. Common dispatcher 24 may process the interrupt itself, e.g., to function as a single stack interrupt handler. Additionally or alternatively, dispatcher 24 may identify the source of the exception (FIQ or IRQ) by examining the Current Processor Status Register (CPSR) mode bits. Once this determination has been made, dispatcher 24 may appropriately branch to either FIQ handler 18' or IRQ handler 16' for further processing in a conventional manner.

Advantageously, this embodiment of the present invention provides a single interrupt stream for both IRQ

and FIQ exceptions, to prevent the prior art occurrence of IRQ exception handling being interrupted by FIQ exceptions. This advantage is particularly apparent in environments in which a single interrupt stack is required. The invention also advantageously permits both the FIQ and IRQ external inputs (not shown) of the ARM® processor to be used by the operating system. This enables users, such as manufacturers of embedded applications, to utilize both IRQ and FIQ pins of ARM® processors, to potentially reduce parts count for advantageous cost savings.

An embodiment of the present invention has been described herein for use with ARM® processors. The skilled artisan will recognize however, that the present invention may be implemented in substantially any processor environment in which: an IRQ vector register precedes an FIQ vector in an exception vector table; it is possible to identify the type of exception; at least one instruction may be processed without interruption; and all interrupts are maskable (i.e., may be shut off) in a single instruction; without departing from the spirit and scope of the invention.

Having thus described the invention, what is claimed is: